Jurnal TICOM: Technology of Information and Communication
Volume 12, Nomor 3, Mei 2024 Hal: 86-90 E-ISSN: 2962-7982

Survei Terhadap Teknik-Teknik *Load Balancing*

Krisna Adiyarta[1,2]

[1]Magister Sistem Informasi Manajemen, Institut Sains dan Bisnis, Pangkalpinang, Indonesia
[2]Fakultas Teknologi Informasi, Universitas Budi Luhur, Jakarta, Indonesia
Jl. Raya Ciledug, Petukangan Utara, Pesanggrahan, Jakarta Selatan, 12260
E-mail: [1]krisna@atmaluhur.ac.id, [2]krisna.adiyarta@budiluhur.ac.id

Abstrak— **Penyeimbangan beban (Load balancing) adalah merupakan issu popular dalam domain diskusi system terdistribusi. Sistem terdistribusi merupakan konsep sistem otonom di mana pengalokasian data diposisikan di berbagai titik atau node. Setiap titik akan mendapatkan tugas yang dapat saja dalam pembagian tugas tersebut beban tugas yang didistribusikan tidak tepat. Ketidaktepatan ini mengakibatkan sistem mungkin akan kelebihan beban. Dalam makalah ini, kami menguraikan berbagai teknik penyeimbangan beban (Load balancing) dengan melakukan penyelesaian masalah dengan menerapkan pergerakan titik dan teknik replikasi. Makalah ini juga mempertimbangkan beberapa teknik pendistribusian beban dengan memperhatikan titik yang memiliki popularitas tinggi sehingga menjadi kelebihan beban kepada titik yang kekurangan beban. Makalah juga menguraikan konsep-konsep baru yang dikembangkan dan penting untuk beberapa teknik penyeimbangan beban dalam lingkungan sistem terdistribusi.**

Kata Kunci— **Load Balancing, Beban Statis, Beban Dinamis, Threshold, Under Loaded Node, Overloaded Node.**

Abstract— ***Load balancing is a popular issue in the distributed system discussion domain. Distributed systems are an autonomous system concept where data allocation is positioned at various points or nodes. Each point will receive a task, where the task load may be distributed incorrectly. This inaccuracy means the system may be overloaded. In this paper, we describe various load balancing techniques by solving the problem by applying point movement and replication techniques. This paper also considers several load distribution techniques by paying attention to points that have high popularity so that they become overloaded to points that are underloaded. The paper also outlines newly developed concepts that are important for several load balancing techniques in distributed system environments.***

Keyword— ***Load balancing, Static load, Dynamic Load, Threshold, Under loaded node, Overloaded node***

I. PENDAHULUAN

Penyeimbangan beban (*Load balancing*) adalah merupakan issu popular dalam domain diskusi system terdistribusi. Sistem terdistribusi merupakan konsep sistem otonom di mana pengalokasian data diposisikan di berbagai titik atau titik pemroses. Setiap titik akan mendapatkan tugas yang dapat saja dalam pembagian tugas tersebut beban tugas yang didistribusikan tidak tepat. Sejumlah teknik telah diajukan dalam upaya untuk penyeimbangan beban yang tepat [1] [2]. Makalah ini juga mempertimbangkan beberapa teknik pendistribusian beban dengan memperhatikan titik yang memiliki popularitas tinggi sehingga menjadi kelebihan beban kepada titik yang kekurangan beban [3] [4]. Konsep Teknik Penyeimbangan beban (*Load balancing*) sebenarnya topik diskusi yang sudah cukup lama namun perkenbangan dan kebermanfaatannya masih dirasakan hingga saat ini. Penyeimbangan beban (*Load balancing*) diterapkan yang tepat dapat dapat dilakukan di mana satu atau banyak cluster dibentuk untuk menyeimbangkan beban. Pemanfaatan ambang batas maksimum, minimum dan normal penting dipertimbangkan untuk penyeimbangan beban yang tepat.

Lingkungan untuk meningkatkan beban dan meningkatkan ketersediaan aplikasi. Di dalam konsep jaringan, penyeimbang beban berlaku kepada interaksi antara mesin klien dan server yang mengatur lalu lintas masuk dan mengarahkan kepada server setelah mempertingkankan dengan menghitung meggunakan berbagai algoritma, Penyeimbang beban berdampak kepada pengurangan beban server individu dengan menyelamatkan titik yang kelebihan beban dan juga akan mencegah kegagalan fungsi pada titik tertentu. Dengan demikian penyeimbangan beban secara langung akan meningkatkan performansi sistem secara keseluruhan. Sebaliknya jika penyeimbangan beban tidak dilakukan dengan benar maka system akan menghadapi masalah yang sangat besar dalam jaringan.

Susunan makalah ini adalah sebagai berikut: Bagian II menjelaskan skema penyeimbangan beban statis, bagian III menjelaskan beberapa pendekatan penyeimbangan beban dinamis dan terakhir bagian IV menyimpulkan makalah.

I. *STATIC LOAD BALANCING*

Penyeimbangan beban secara statis (Static Load Balancing) [5] [6] seluruh informasi dalam melakukan penyeimbangan beban diberikan terlebih dahulu kepada system. Kinerja proses dihitung pada saat eksekusi. Informasi beban dihitung pada satu titik pemroses kemudian dikirimkan untuk dieksekusi di titik pemroses secara jarak jauh. Tergantung kepada perhitungan beban kerja secara terdistribusi, alokasikan beban dan pelaksanaannya tidak memperhatikan beban sebelumnya. Bentuk non-preemptive diperoleh dengan penyeimbangan beban secara statis. Perhitungan beban sistem tidak bergantung pada keadaan saat ini akan tetapi perhitungan memperhatikan sumber daya dan bobot dari sistem. Pada algoritma static load balancing, pendistribusian beban dilakukan lebih awal. Terdapat beberapa algoritma penyeimbangan beban statis dalam mendistribusikan beban kerja [4]. Pembahasan terhadap

skema-skema algoritma static load balancing adalah sebagai berikut.

A. Algoritma *Central Manager*

Dalam Algoritma *Central Manager* penyeimbangan beban, sebuah titik pemroses dipilih untuk dijadikan titik pemroses pusat. Pemilihan berdasarkan kepada seberapa banyak permintaan yang dapat diterima. Setelah memilih titik pemroses pusat untuk mengelola permintaan tersebut. Pemindahan beban dari titik pemroses yang kelebihan beban ke titik pemroses yang ringan dan membuat jalur yang tepat untuk menentukan pekerjaan utama dilakukan oleh titik pemroses pusat. Algoritma ini tidak diterapka pada model proses terdistribusi karena diperlukan titik pemroses pusat untuk pengelolaan fungsi.

B. Algoritma *Round Robin*

Penyeimbangan beban *Round Robin* [7] merupakan salah satu metode paling sederhana di mana informasi didistribusikan oleh klien ke sekelompok server. Teknik ini diterapkan dalam jaringan dengan jumlah server terbatas. Ketika klien melakukan permintaan untuk diteruskan ke server, server pertama mengalokasikan permintaan tersebut. Ketika ada permintaan kedua, permintaan tersebut dialokasikan ke server kedua. Manakala permintaan ketiga datang ke server ketiga permintaan tersebut dialokasikan ke server ketiga. Tetapi jika server menerima permintaan dan seluruh server sedang dalam pelaksanaan pekerjaan maka permintaan tersebut dialokasikan ke *server* pertama dan akan membentuk alokasi dengan bertipe *Ring*.

Terdapat dua jenis algoritma *Round Robin* yaitu *: Classic Round Robin Algorithm* dan *Weighted Round Robin Algorithm*. Dalam algoritma *Round Robin* Klasik pendekatan irisan waktu tertentu yang diberikan untuk setiap proses. Sedangkan dalam algoritma *Round Robin* berbobot, setiap titik pemroses telah ditetapkan nilai bobot. Pemberian bobot didasarkan kepada konfigurasi dan kriteria untuk memproses permintaan. Bobot pada titik pemroses akan menentukan waktu proses yang dapat diterapan pada titik pemroses yang bersangkutan.

C. *Algoritma Threshold*

Dalam algoritma *Threshold* (ambang batas), titik pemroses diberikan bobot dan setiap titik pemroses dapat menentukan nilai ambang batas. Nilai ambang batas akan menentukan kapasitas maksimum titik pemroses dalam menerima beban kerja. Tiga Tingkat kondisi server yang dapat terjadi pada setiap titik pemroses yaitu: *Underloaded*, *Normal* dan *Overloaded* di mana

if load< threshold (Underloaded)
load>threshold (overloaded)
load <=threshold (Normal)

Pada tingkat awal semua titik pemroses berada dalam kondisi kekurangan beban, ketika status beban melewati batas ambang batas maka titik pemroses disebut pada tingkat kelebihan beban. Ketika titik pemroses mencapai batas ambang batas, titik pemroses tersebut tidak akan menerima permintaan lainnya.

D. Algoritma *Min-Min Load Balancing*

Dalam algoritma *Min-Min Load Balancing* ini, semua informasi terkait dengan beban kerja sudah ditentukan sebelumnya. Didasarkan atas informasi dari semua pekerjaan maka waktu eksekusi dari setiap proses dihitung. Pekerjaan dengan waktu eksekusi minimum dipilih untuk kemudian ditentukan titik pemroses sebagai pelaksana perkerjaan tersebut. Pekerjaan diproses sesuai dengan waktu pemilihan hingga semua pekerjaan tereksekusi.

E. Algoritma *Min-Max Load Balancing*

Algoritma ini akan bekerja berlawanan [7] dibandingkan dengan algoritma *min-min* dimana titik pemroses minimum dieksekusi terlebih dahulu. Dalam algoritma ini, setelah menemukan waktu penyelesaian rata-rata dari semua titik pemroses dihitung, kemudian titik pemroses yang memiliki waktu penyelesaiannya lebih besar dari nilai rata-rata dipilih dan pekerjaan tersebut akan diproses ulang ke titik pemroses yang memiliki waktu penyelesaian paling sedikit.

F. *Algoritma Randomized*

Dalam Algoritma *Randomized*, pemilihan titik pemroses dilakukan secara acak bukan didasarkan atas satu basis tertentu. Server mengirimkan permintaan ke sebuah titik pemroses dan kemudian permintaan dapat diproses ke titik pemroses tersebut. Operasi ini dilakukan oleh *load balancer*. Masalah utama dalam teknik ini adalah masalah kelebihan beban Dimana mungkin saja terjadi sebuah titik pemroses menerima kelebihan beban.

III. DYNAMIC LOAD BALANCING

Dalam teknik ini, banyak titik pemroses bertanggung jawab untuk membuat keputusan penyeimbangan beban yang benar. Dalam pendekatan dinamis [3] [4] [5] [7] simpulnya tidak pasti dan permintaannya tidak sederhana. Suatu saat beban akan berkurang dan di lain waktu load akan tinggi. Oleh karena itu, kami telah menetapkan algoritma penyeimbangan dinamis yang memberikan hasil yang benar dalam lingkungan run time. Setiap titik pemroses berisi informasi titik pemroses satu sama lain atau titik pemroses tetangga. Kegagalan sebuah titik pemroses tidak diabaikan dalam skenario terdistribusi, kami memiliki konsep baru untuk menjadikan sistem sebagai toleransi kesalahan. Jadi dalam pendekatan hari ini kami melakukan penyeimbangan beban dinamis, bukan penyeimbangan beban statis. Pembahasan beberapa algoritma dinamis adalah sebagai berikut:

A. Algoritma *Nearest Neighbor* (NNA)

Pada NNA, setiap titik pemroses melakukan algoritma penyeimbangan beban. Pada algoritma NNA setiap titik pemroses mencari titik pemroses lain yang kelebihan beban jika ditemukan kemudian akan memigrasikan permintaan sebaliknya jika tidak ditemukan titik pemroses akan mengirimkan beban ke tetangganya sendiri. Memuat informasi

dipertukarkan secara lokal berarti kami telah melakukan pekerjaan di lingkungan terbatas. Dalam algoritma ini, semua titik pemroses memperhatikan titik pemroses yang berdekatan dan distribusi beban ke semua titik pemroses secara merata. Agar distribusi beban lokal berhasil, penyeimbang beban global dipertimbangkan.

B. Algoritma *Threshold and Least*

Pada algoritma *Threshold and Least* [4], ini merupakan dua algoritma yang berbeda namun menggunakan pendekatan yang serupa. Dalam algoritma *Threshold*, jika beban berada di bawah ambang batas tertentu maka menendakan bahwa sebuah titik pemroses telah menerima beban dari titik pemroses lain yang kelebihan beban. Apabila tidak ada titik pemroses yang tersedia untuk menerima beban maka proses dijalankan secara lokal. Algoritma *Least* merupakan bagian dari algoritma *Threshold*. Teknik ini menghitung waktu respons paling sedikit titik pemroses mana yang memiliki waktu respons rendah akan sangat aktif menerima beban. Komunikasi ini bersifat lokal. Kedua algoritma ini sederhana dan efektif untuk diterapkan pada lingkungan yang tidak terlalu sibuk.

C. Algoritma *Random* (RAND)

Dalam Algoritma *Random* (RAND) [3], apabila terdapat titik pemroses dengan beban kerja (yang lebih besar dari batas ambang batas) yang teridentifikasi oleh titik pemroses mana pun, maka beban tersebut akan dimigrasikan ke titik pemroses lain yang dipilih secara acak dengan tidak memperhatikan informasi titik pemroses, apakah titik pemroses tersebut akan kelebihan beban atau kekurangan beban. Algoritma ini tidak menyimpan informasi beban lokal apa pun atau mengirimkan informasi beban apa pun ke titik pemroses lain. Selain itu, desainnya sederhana dan implementasinya mudah diterapkan. Pertukaran informasi diatur secara lokal dan setiap titik pemroses akan dipilih melalui proses sacara acak. Bagian penting dalam algoritma ini adalah dimana beban didistribusikan secara *asynchronous*.

D. Algoritma *Adapting Contracting with Neighbor* (ACWN)

Dalam algoritma ini ada pemanfaatan tabel yang akan diperbarui setiap kali ada pengiriman/pemindahan beban. Algoritma ACWN perlu menjaga informasi beban lokal dan juga informasi beban titik pemroses tetangga yang melakukan pertukaran beban dari waktu ke waktu [3]. Algortima ini akan secara otomatis menemukan titik pemroses tujuan yang lebih memiliki beban yang sedikit. Maka, ACWN mengelola dengan pendekatan penyeimbangan beban dinamis dan dengan melakukan pertukaran informasi berbasis lokal.

E. Algoritma *Never Queue*

Pada algoritma ini [8], pekerjaan dikirim ke setiap server tujuan melalui server pengirim dan perhitungan biaya dilakukan. Cara kerja utama dari algoritma ini adalah permintaan dapat segera diproses dan tidak perlu menunggu berdasarkan antrian. Teknik ini meminimalkan waktu tunda pada proses selanjutnya, sehingga hasilnya adalah meminimalisir penundaan dari keseluruhan proses. Namun kelemahan dari algoritma ini adalah server tidak memproses pekerjaan masuk sampai server yang lebih cepat tersedia untuk memproses pekerjaan secara efisien. Jadi algoritma never queue bagus untuk respon cepat.

F. Algoritma *Cyclic*

Pada algoritma penyeimbangan beban yang berbasis *Cyclic* [4] beban ditetapkan dalam suatu sistem secara melingkar. Kelebihan dari algoritma *Cyclic* ini adalah di mana proses yang menerima informasi tidak akan menerima permintaan untuk waktu tertentu, kecuali siklus tersebut tidak dapat terselesaikan untuk satu putaran. Informasi disimpan oleh titik pemroses Dimana merupakan sumber dari proses. Oleh karena itu apabila sumbernya gagal maka seluruh informasi dikirim kembali ke titik pemroses tertentu. Informasi dalam algoritma ini dipertukarkan berbasis lokal.

G. *Probabilistic*

Algoritma berbasis probabilistik, informasi terkait setiap titik pemroses dipertahankan dan sesuai dengan informasi titik pemroses tersebut diberikan probabilitas. Berdasarkan probabilitas, titik pemroses yang probabilitasnya lebih tinggi akan menerima informasi dari titik pemroses yang probabilitasnya lebih kecil, artinya titik pemroses yang memiliki probabilitas lebih tinggi akan menerima beban dari titik pemroses yang memiliki probabilitas tinggi. Untuk mencari probabilitas, informasi tentang setiap titik pemroses dihitung dengan rumus beban/beban total.

H. Algoritma *Pipelining*

Algoritma ini didasarkan pada *pipelining*. Output tahap pertama akan difungsikan sebagai input untuk tahap kedua, oleh karena itu dibuatlah sebuah pipa yang disebut pipa semu (*Virtual Pipe*). Dalam pendekatan berbasis *pipelining* [5] jika beban pertama diproses maka kita memulai proses kedua tanpa mengganggu yang pertama.

I. Algoritma *Prioritized Random*

Dalam penyeimbangan beban acak dengan prioritas [5], sistem harus memilih beban titik pemroses secara acak yang memiliki titik pemroses kelebihan beban seperti algoritma acak. Dalam PRAND, prioritas diberikan pada setiap titik pemroses di mana, titik pemroses yang memiliki prioritas lebih tinggi akan mengirimkan beban ke titik pemroses lain yang memiliki prioritas lebih rendah.

J. Algoritma *Reception*

Dalam algoritma berbasis *Reception* [8], titik pemroses yang mempunyai nilai di bawah ambang batas akan mancari titik pemroses yang kelebihan beban secara acak atau dengan menggunakan pendekatan polling dan mentransfer beban dari titik pemroses yang kelebihan beban ke titik pemroses yang bebannya ringan. Algoritma ini mirip dengan pendekatan berbasis penerima dan mentransfer informasi secara lokal.

K. Algoritma *Centralized Information dan Centralized Decision*

Dalam algoritma informasi terpusat (*Centralized Information*) [4], informasi tersedia pada satu titik pemroses.

Seluruh titik pemroses lainnya berkomunikasi dengan titik pemroses pusat, dan keputusan didasarkan pada pendekatan berbasis tunggal. Titik pemroses pusat adalah titik pemroses utama dari algoritma. Ketika titik pemroses yang kelebihan muatan ingin mentransfer beban, ia meminta server pusat kemudian server pusat tersebut mencari titik pemroses yang memiliki beban ringan. Setiap titik pemroses dalam sistem menghubungkan mesin server pusat sehingga server bertanggung jawab untuk mentransfer titik pemroses dan keputusannya juga terpusat.

L. Algoritma *The Shortest Expected Delay* (SED)

Algoritma berbasis Strategi *Shortest Expected Delay* (SED)[5] didasarkan pada waktu tunda yang diharapkan. Setelah setiap pekerjaan selesai, nilai penundaan dihitung dari titik pemroses sumber ke titik pemroses tujuan dan pilih titik pemroses yang penundaan titik pemrosesnya minimal. Algoritma ini mirip dengan *Greedy* Algorithm. Dalam pendekatan ini, setiap titik pemroses menjalankan peran sesuai dengan fungsi terbaiknya dan disatukan dalam antrian yang memiliki perkiraan waktu tunda minimum dalam setiap penyelesaian beban.

M. Algoritma *Tilling* ataur *Direct Neighborhood*

Dalam algoritma *Tilling* atau *Direct Neighborhood* [4] pertukaran informasi bersifat lokal. Saat melakukan distribusi beban, penyeimbangkan proses dilakukan dalam lingkungan yang sama. Beban didistribusikan di antara proses dalam lingkungan yang sama. Komunikasi antar proses bersifat lokal dan semua tugas akan dijalankan secara independen. Dalam algoritma ini sistem memiliki titik pemroses keseimbangan untuk mentransfer tetangga secara langsung di lingkungan yang sama.

N. Algoritma *X-Tiling*

Dalam pendekatan ini sistem mempertimbangkan *hypercube*. Prosesor penyeimbang terhubung dalam hypercube [5]. Beban didistribusikan ke seluruh proses di *hypercube*. Komunikasi titik pemroses seragam secara global dan inisialisasi algoritma dilakukan secara berkala. Pendekatan *X-Tilling* adalah pertukaran informasi secara lokal. Algoritma ini berbasis sinkronisasi artinya system melakukan sinkronisasi titik pemroses dengan cara *hypercube*.

N. Algoritma *Throttled Based*

Dalam *Throttled Load Balancer* (TLB), system akan menyimpan catatan status di setiap titik pemroses dan membuat tabel. Pada saat permintaan dikirimkan, ia mencari titik pemroses yang telah ditentukan sebelumnya dalam tabel dan jika kecocokan ditemukan berdasarkan ketersediaan titik pemroses dan juga titik pemroses yang dipilih memiliki ukuran yang sama, maka permintaan tersebut diberikan, jika tidak, nilai negasi akan dikembalikan dan permintaan akan dimasukkan ke dalam antrian Kembali.

O. Algoritma *Centralized Information* dan *Distributed Decision*

Dalam informasi berbasis pusat (*Centralized Information*) [4] informasinya terpusat tetapi keputusan dari titik pemroses didistribusikan. Dalam pendekatan ini, server menginformasikan tentang beban dari setiap titik pemroses. Setiap titik pemroses menerima informasi satu sama lain. Informasi tentang titik pemroses yang kelebihan beban dapat ditemukan dengan mudah dan transfer beban dari titik pemroses yang memiliki beban berat ke titik pemroses ringan dapat dilakukan tanpa mengganggu server. Algoritma ini melakukan pekerjaan secara efisien karena pertukaran informasi pesan yang lebih sedikit. Sifatnya juga kuat karena kegagalan titik pemroses pusat tidak mempengaruhi algoritma untuk melakukan operasi beban dengan sempurna.

P. Algoritma *Divide* dan *Conquer*

Dalam pendekatan ini, kita membagi beban dari proses kemudian melakukan operasi beban. Setelah melakukan pemuatan, sistem menggabungkan dua titik pemroses kemudian titik pemroses ini melakukan penyeimbangan beban. Pendekatan yang sama akan diterapkan untuk semua titik pemroses dan terakhir sistem akan menemukan titik pemroses yang digabungkan sepenuhnya dan akhirnya semua titik pemroses seimbang.

Q. Algoritma *Dynamic Ratio*

Serupa dengan pendekatan *Static Ratio* dimana dalam menentukan rasio kinerja server mana dan berapa banyak kapasitas memori yang disediakan. Pendefinisian dilakukan secara statis yang dilakukan secara manual di setiap rasio server. Dalam metode *Dynamicc Ratio* [9] pendefinisian setiap server memperhatikan kapasitas memori yang berbeda dan semua server dikelola oleh unit pusat yang disebut *load balancer*. Di penyeimbang beban, semua informasi disimpan dalam tabel. Dalam *Dynamicc Ratio* pendefinisian rasio dilakukan secara dinamis artinya setiap kali melakukan memproses permintaan, rasio akan diubah.

IV. KESIMPULAN

Makalah ini menyajikan survei tentang beberapa algoritma penyeimbangan beban dalam lingkungan sistem terdistribusi. Sistem terdistribusi adalah kumpulan titik pemroses di mana banyak tugas diserahkan pada titik pemroses berbeda. Karena tugas yang datang secara acak pada titik pemroses yang berbeda, ada kemungkinan bahwa hanya sedikit titik pemroses yang memuat banyak dan sedikit di antaranya yang memuat sedikit. Makalah ini menyajikan beberapa pendekatan penyeimbangan beban baik dalam pendekatan statis maupun dinamis.

REFERENSI

[1] N. Rathore, and I. Chana, "Load Balancing and Job Migration Techniques in Grid: A Survey of Recent Trends," *In Wireless Personal Communication: An International Journal*, vol. 79, issue. 3, pp. 2089-2125, 2014. https://doi.org/10.1007/s11277-014-1975-9

[2] A. Soltani, and M. Sharifi, "A Load Balancing Algorithm Based on Replication and Movement of Data Items for Dynamic Structured P2P System," *International Journal of Peer to Peer Networks* (IJP2P), vol.5, no.3, pp. 15-32, 2014

[3] P. B. Soundarabai, et al, "Comparative Study of Load Balancing Techniques in Distributed System," *International Journal of*

Information Technology and Knowledge Management, vol. 6, no. 1, pp. 53-60, 2012.

[4] M. Z. Khan, R. Singh, J. Alam, and S. Saxena, "Classification of Load Balancing Condition for Parallel and Distributed System," *IJCSI: International Journal of Computer Science Issues*, vol. 8 issue. 5, pp. 411-419, 2011.

[5] M. Kumari, and R. K. Katare, "A Comparative Study of Various Load Balancing Algorithm in Parallel and Distributed Multiprocessor System," *International Journal of Computer Applications*, vol. 169, no. 10, pp. 31-35, 2017

[6] Z. M. Elngomi, and K. Khanfar, "A Comparative Study of Load Balancing Algorithms: A Review Paper," *IJCSMC*, vol. 5, issue. 6, pp.448–458, 2016.

[7] N. Mishra, and N. K. Mishra, "Load Balancing Techniques: Need, Objectives and Major Challenges in Cloud Computing- A Systematic Review," *International Journal of Computer Applications*, vol. 131, no. 18, pp.11-19, 2015.

[8] Dr. C. Mukundha, N. Venkatesh and K. Akshay, "A Comprehensive Study Report on Load Balancing Techniques in Cloud Computing," *International Journal of Engineering Research and Development*, vol. 13, issue. 9, pp. 35-42, 2017.

[9] P. B. Soundarabai, et al, Comparative Study on Load Balancing Techniques in Distributed System," *International Journal of Information Technology and Knowledge Management*, vol. 6, no. 1, pp. 53-60, 2012.